Traduction Automatique Neuronale avec OpenNMT

Buniatyan Galust
Salloum Mark

1 Introduction

La traduction automatique est un domaine de recherche qui consiste à faire traduire automatiquement du texte d'une langue à une autre par un programme informatique. Au fil du temps, différentes approches se sont succédées : les méthodes basées sur des règles linguistiques, puis les modèles statistiques, et plus récemment les modèles neuronaux. Ces derniers, dits de traduction automatique neuronale, ont connu un essor considérable grâce aux progrès de deep learning et à la disponibilité de ressources massives (corpus parallèles, puissance de calcul GPU).

Dans le cadre de ce projet, nous nous intéressons à la traduction automatique neuronale en utilisant OpenNMT. L'objectif est de mettre en place et d'évaluer différents modèles de traduction neuronale, d'abord sur des corpus dits « en formes fléchies » (c'est-à-dire tels qu'on les rencontre naturellement), puis sur des corpus « lemmatisés » (où chaque mot est remplacé par son lemme). Nous analyserons ensuite dans quelle mesure ces deux versions du corpus influent sur les performances de traduction. Les évaluations seront faites à l'aide de la métrique BLEU, largement utilisée pour comparer une hypothèse de traduction à une référence humaine.

2 Présentation du moteur de traduction neuronale OpenNMT

2.1 Fonctionnement général d'OpenNMT

OpenNMT est un outil open-source créé pour faciliter la construction et l'entraînement de modèles de traduction automatique neuronale. OpenNMT fournit :

- Un ensemble de scripts et d'API pour prétraiter les données (construction du vocabulaire, tokenisation, etc.).
- Des modèles pré-implémentés, principalement basés sur des réseaux de neurones récurrents (RNN) ou sur l'architecture Transformer (plus récente).

— Des options de configuration flexibles pour l'entraînement (taille des batchs, nombre d'époques, taux d'apprentissage, etc.).

Le fonctionnement d'OpenNMT suit généralement les étapes suivantes :

1. Préparation du corpus : séparation des données en entraînement (TRAIN), développement (DEV) et test (TEST), éventuellement nettoyage et tokenisation.
2. Construction du vocabulaire : création de listes de tokens (mots ou sous-mots) pour la langue source et la langue cible.
3. Entraînement du modèle : un réseau neuronal (RNN/LSTM ou Transformer) apprend à mapper les séquences de la langue source vers la langue cible.
4. Traduction : lors de l'inférence, un algorithme de recherche génère la séquence cible à partir de la séquence source.
5. Évaluation : on compare la traduction générée à une référence humaine à l'aide de métriques telles que BLEU.

2.2 Approches de modélisation : LSTM ou Transformers

OpenNMT prend en charge plusieurs architectures, dont les RNN (Réseaux de Neurones Récurrents) avec des unités LSTM ou GRU, et l'architecture Transformer.

Modèle RNN/LSTM :

— L'encodeur lit la phrase source token par token et encode l'information dans ses états cachés.

— Le décodeur génère la phrase cible progressivement, en tenant compte du contexte transmis par l'encodeur et du mot précédemment généré.

Modèle Transformer :

— Il repose sur un mécanisme d'attention multi-têtes (self-attention) qui permet au réseau de traiter l'intégralité de la séquence en parallèle, sans passer par une récursivité explicite.

— Les Transformers se sont imposés comme l'architecture de référence dans la plupart des travaux récents de traitement automatique des langues (notamment BERT, GPT, etc.).

— Ils sont généralement plus rapides et donnent de meilleurs résultats, au prix d'avoir besoin de plus de données et de ressources GPU.

3 Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en formes fléchies

3.1 Corpus d'apprentissage et d'évaluation

Pour nos expériences, nous avons retenu deux corpus parallèles (anglais-français) :

— Europarl (domaine parlementaire / général)

— Emea (domaine médical)

Les fichiers sources (.en) et cibles (.fr) ont été nettoyés et tokenisés. Puis, conformément aux consignes, nous avons extrait :

— Europarl :
  — Train : 100 000 premières phrases
  — Dev : 3 750 phrases suivantes
  — Test : 500 phrases supplémentaires
— Emea :
  — Train : 10 000 premières phrases
  — Test : 500 phrases supplémentaires

Le prétraitement comprenait la **tokenisation** (scripts de Moses) et le **nettoyage** (limitant les phrases à 80 tokens). Au total, nous avons donc obtenu des fichiers *.clean.en et *.clean.fr alignés et prêts pour OpenNMT.

3.2 Métrique d'évaluation : Score BLEU

Pour évaluer la qualité des traductions générées, nous utilisons la métrique BLEU (Bilingual Evaluation Understudy). BLEU compare les n-grammes de la traduction produite à ceux d'une traduction de référence humaine. Nous faisons appel à l'outil SacreBLEU via la commande :

```
sacrebleu reference.fr -i prediction.fr -m bleu -b --force
```

3.3 Résultats expérimentaux

Nous avons réalisé deux entraînements (ou runs) :

— Run 1 : entraînement sur Europarl seul (100 k phrases).
— Run 2 : entraînement sur Europarl (100 k) + Emea (10 k).

Dans les deux cas, l'ensemble de d'apprentissage est suivi d'un ensemble de développement de 3 750 phrases Europarl pour valider la convergence. Nous testons finalement sur 500 phrases Europarl (in-domain) et 500 phrases Emea (out-of-domain).

Après avoir construit le vocabulaire (`onmt_build_vocab`) et entraîné le modèle (`onmt_train`) pour 10 000 pas (`train_steps: 10000`), nous avons généré des traductions avec `onmt_translate`. Les scores BLEU sont calculés grâce à `sacrebleu`.

Les résultats sont synthétisés ci-dessous :

Modèle	Europarl Test	Emea Test
Run 1 (Europarl 100 k)	15.7	1.4
Run 2 (Europarl 100 k + Emea 10 k)	17.5	5.8

TABLE 1 – Scores BLEU sur Europarl_test_500 (in-domain) et Emea_test_500 (out-of-domain).

3.4 Discussion

Les résultats montrent un **gain modéré sur Europarl (in-domain)** et un **gain plus marqué sur Emea (hors-domaine)** :

- **Europarl_test_500** : Le modèle entraîné uniquement sur Europarl (Run 1) obtient un BLEU de 15.7, contre 17.5 lorsqu'on ajoute 10 000 phrases Emea (Run 2).
  - L'amélioration demeure limitée, probablement parce que 10 k de données médicales n'apportent qu'un léger complément de diversité linguistique. Une partie du vocabulaire Emea, certes différente, a pu cependant bénéficier à une meilleure généralisation sur des textes parlementaires.
  - On pourrait s'attendre à un écart plus prononcé si on incorporait un volume plus important de données Emea, ou encore un autre corpus du même domaine que Europarl.
- **Emea_test_500** : Sur le hors-domaine, on constate un passage de BLEU=1.4 à BLEU=5.8, ce qui confirme l'importance de disposer de données spécialisées.
  - Bien que 5.8 reste faible en absolu, on note tout de même une amélioration significative par rapport à 1.4.
  - Ce score relativement bas suggère que 10 000 phrases Emea ne suffisent pas pour couvrir la variété terminologique du domaine médical. Il est probable qu'un corpus Emea plus vaste (50 k, 100 k, etc.) permettrait au modèle d'acquérir un vocabulaire et des structures plus adaptées, rehaussant encore la performance.

En résumé, ajouter 10 k phrases Emea dans le mélange d'entraînement permet de mieux couvrir le vocabulaire médical et d'éviter un effondrement complet hors-domaine, tout en améliorant légèrement la performance in-domain. Néanmoins, la taille limitée du corpus Emea explique que le gain demeure relativement modeste sur Europarl (17.5 contre 15.7), et encore insuffisant pour obtenir un BLEU satisfaisant sur Emea test (5.8). Cela illustre bien à quel point la quantité et la proximité thématique des données d'entraînement ont un impact direct sur la qualité de la traduction neuronale.

4 Évaluation du moteur de traduction neuronale OpenNMT sur un corpus en lemmes

Dans cette section, nous répliquons le même protocole que pour les corpus en formes fléchies, mais en appliquant cette fois une **lemmatisation** aux corpus (anglais et français). Nous décrivons d'abord les lemmatiseurs utilisés, puis présentons les résultats obtenus lorsque nous ré-entraînons OpenNMT sur ces corpus lemmatisés.

4.1 Lemmatiseurs utilisés

Initialement, nous avions adopté une approche basée sur NLTK (WordNet-Lemmatizer) pour l'anglais et FrenchLefffLemmatizer pour le français, conformément aux consignes du projet. Cependant, cette méthode présentait plusieurs limitations : une lenteur notable dans le traitement des grands corpus (environ 6 minutes pour 100 000 phrases), une analyse contextuelle limitée, et surtout, nous avions commis l'erreur méthodologique de lemmatiser les fichiers déjà nettoyés (.clean) plutôt que d'appliquer la lemmatisation avant le nettoyage. Face à ces défis, nous avons décidé de reconsidérer notre approche, ce qui nous a conduits à adopter spaCy comme alternative plus performante et à revoir notre séquence de traitement.

4.1.1 Utilisation de spaCy pour la lemmatisation

Pour améliorer la qualité et l'efficacité de la lemmatisation, nous avons opté pour la bibliothèque spaCy, reconnue pour ses performances et sa robustesse dans le traitement du langage naturel. Notre choix s'explique par plusieurs avantages :

- **Analyse contextuelle** : Contrairement aux approches précédentes, spaCy prend en compte le contexte complet de la phrase lors de la lemmatisation
- **Performance optimisée** : Son traitement par lots permet une lemmatisation beaucoup plus rapide sur de grands corpus
- **Richesse linguistique** : Les modèles pré-entraînés pour l'anglais (`en_core_web_sm`) et le français (`fr_core_news_sm`) offrent une couverture lexicale plus complète
- **Désactivation sélective** : Nous pouvons conserver uniquement les composants nécessaires à la lemmatisation, accélérant ainsi le traitement

Notre implémentation utilise une approche optimisée avec :

- Traitement par lots (*batch processing*) pour maximiser l'efficacité
- Désactivation des composants non essentiels (comme le NER et le parser)
- Barre de progression pour surveiller le traitement des corpus volumineux

4.2 Processus de traitement

Notre méthodologie se distingue par l'ordre des opérations que nous avons adopté :

1. **Lemmatisation d'abord** : Nous appliquons la lemmatisation sur les fichiers d'origine
2. **Nettoyage ensuite** : Les fichiers lemmatisés sont ensuite nettoyés avec les scripts Moses

Cette séquence est fondamentale car :

- La lemmatisation fonctionne mieux sur des textes naturels non nettoyés où le contexte est entièrement préservé
- Les lemmatiseurs peuvent ainsi s'appuyer sur tous les indices grammaticaux et contextuels présents dans le texte d'origine

— Le nettoyage peut éliminer des indices précieux pour la lemmatisation s'il est appliqué en premier
— En lemmatisant d'abord, nous évitons que le processus de nettoyage n'altère des relations lexicales importantes

Cette approche nous a permis d'obtenir une meilleure qualité de lemmatisation tout en préservant l'intégrité des données pour les étapes ultérieures.

4.3 Résultats sur corpus lemmatisés

Comme précédemment, nous évaluons chacun des modèles :
— In-domain : Europarl_test_500.lem.clean (500 phrases lemmatisées puis nettoyées)
— Out-of-domain : Emea_test_500.lem.clean (500 phrases lemmatisées puis nettoyées)

Les scores BLEU, calculés via `sacrebleu`, sont récapitulés ci-dessous :

Modèle (lemmatisé avec spaCy)	Europarl (in-domain)	Emea (out-of-domain)
Run "Europarl lemmatisé"	17.7	4.4
Run "Europarl + Emea lemmatisé"	20.9	9.1

TABLE 2 – Scores BLEU sur Europarl_test_500.lem.clean et Emea_test_500.lem.clean.

Analyse des résultats
— **Sur Europarl (in-domain)** : Le modèle "Europarl lemmatisé" atteint BLEU = 17.7, tandis que le modèle "Europarl + Emea lemmatisé" obtient 20.9. Nous observons une amélioration significative par rapport à nos expériences précédentes, avec un gain de 3.2 points BLEU lorsque les données Emea sont incluses.
— **Sur Emea (hors-domaine)** : Le modèle "Europarl lemmatisé" obtient BLEU = 4.4, ce qui représente déjà une amélioration notable par rapport aux approches précédentes. Le modèle "Europarl + Emea lemmatisé" atteint 9.1, confirmant l'importance combinée d'une bonne lemmatisation et de l'inclusion de données spécifiques au domaine.

Ces résultats démontrent clairement que l'utilisation de spaCy pour la lemmatisation, combinée à l'approche "lemmatiser d'abord, nettoyer ensuite", apporte des gains substantiels en termes de qualité de traduction. L'amélioration est particulièrement notable pour le modèle mixte (Europarl + Emea), où l'effet est synergique : une meilleure lemmatisation permet au modèle de mieux exploiter les similarités entre les domaines, tandis que l'inclusion de données spécifiques au domaine médical renforce sa capacité à gérer la terminologie spécialisée.

4.4 Conclusion sur la lemmatisation

La transformation des textes en lemmes avec spaCy, suivie d'un nettoyage standard, s'avère nettement plus efficace que nos approches précédentes. Ces résultats confirment l'importance d'un prétraitement linguistique robuste et contextuellement informé.

Le gain de performance observé (de 15.7 à 17.7 pour Europarl seul, et de 17.5 à 20.9 pour le modèle mixte) démontre que la qualité de la lemmatisation impacte directement les performances de traduction. La capacité de spaCy à analyser le contexte et à produire des lemmes plus précis semble donc être un facteur déterminant pour améliorer la généralisabilité des modèles de traduction neuronale.

5 Points forts, limitations et difficultés rencontrées

5.1 Points forts

Workflow complet : Nous avons mis en place une chaîne de traitement complète pour la traduction neuronale : prétraitement des données (tokenisation, nettoyage, lemmatisation), configuration YAML, entraînement OpenNMT et évaluation via la métrique BLEU.

Approche flexible : Le recours à OpenNMT a offert la possibilité de tester différentes configurations, y compris l'entraînement sur corpus mixte (Europarl+Emea) et l'entraînement sur des données lemmatisées.

Environnement accessible (Google Colab) : L'utilisation de Google Colab nous a permis de bénéficier d'un GPU gratuit pour accélérer l'entraînement, rendant le projet réalisable sur un simple navigateur.

Mode "rapide" via Google Drive : Le notebook peut être utilisé dans deux modes :

- **Mode rapide** : Pour éviter les temps de téléchargement, nettoyage et entraînement (qui peuvent prendre environ une heure), il suffit d'utiliser les fichiers déjà présents sur notre Drive (dans le dossier partagé Projet_OpenNMT).
- **Mode complet** : Tout télécharger et nettoyer localement, puis entraîner le modèle.

Ainsi, chacun peut choisir la méthode la plus pratique, en fonction de sa connexion et du temps disponible.

5.2 Limitations

Quantité de données limitées : Nous nous sommes limités à 100 000 phrases Europarl et 10 000 phrases Emea. Or, l'apprentissage profond nécessite souvent bien plus de données pour atteindre des performances élevées, surtout dans des domaines spécialisés comme le médical.

Modèle RNN (LSTM) par défaut : Dans OpenNMT-py, le modèle par défaut décrit dans la documentation (notamment le Quickstart) est un réseau

RNN avec LSTM. Concrètement, lorsque nous ne modifions pas les hyperparamètres du modèle dans le fichier YAML (par exemple, si nous ne spécifions pas `-model_type transformer` ou `-encoder_type transformer`), OpenNMT utilise un réseau RNN à deux couches, à base de LSTM.

Ainsi, en suivant la configuration minimale (quickstart ou YAML par défaut) sans changer le type de réseau, nous avons naturellement entraîné un modèle RNN (LSTM).

Dans la documentation OpenNMT-py, on retrouve :

> "This configuration will run the default model, which consists of a 2-layer LSTM..."

Un modèle Transformer aurait potentiellement pu offrir de meilleures performances (à condition d'avoir plus de ressources ou de temps d'entraînement).

Optimisation de la lemmatisation : Bien que l'utilisation de spaCy ait considérablement amélioré notre processus de lemmatisation par rapport aux approches initiales, certaines limitations persistent. Pour les langues très morphologiquement riches ou dans des domaines hautement spécialisés (comme le médical), même spaCy peut rencontrer des difficultés avec des termes techniques ou rares. Une adaptation supplémentaire des modèles spaCy ou l'utilisation d'un lexique spécialisé pourrait apporter des améliorations additionnelles.

Efficacité du traitement par lots : Malgré l'optimisation du traitement par lots dans notre processus de lemmatisation avec spaCy, la gestion de très grands corpus reste un défi. Pour des ensembles de données plus importants (plusieurs millions de phrases), des stratégies de parallélisation plus avancées seraient nécessaires.

5.3 Difficultés rencontrées

Configuration et dépendances : L'installation et la configuration d'OpenNMT, de Moses, de spaCy et de ses modèles linguistiques ont parfois nécessité plusieurs essais (versions incompatibles, problème d'espace dans Google Colab, etc.). En particulier, l'installation des modèles de langue spaCy de taille moyenne ou grande peut s'avérer problématique dans des environnements à mémoire limitée comme Colab.

Temps d'entraînement : Même si Colab fournit un GPU, l'entraînement peut être long selon la taille des données et la configuration du modèle. Nous avons dû ajuster le nombre de pas (`train_steps`) et surveiller la mémoire.

Gestion de l'alignement : Il était crucial de respecter l'alignement des phrases entre fichiers .en et .fr lors des différentes étapes de traitement (lemmatisation puis nettoyage), sous peine de fausser les résultats. Maintenir cet alignement tout au long du processus en deux étapes a nécessité une attention particulière.

Équilibre entre qualité et performance : Trouver le bon équilibre entre la qualité de la lemmatisation (en désactivant ou non certains composants de spaCy) et la vitesse de traitement a représenté un défi. Nous avons dû expérimenter avec différentes configurations pour déterminer quels composants

étaient vraiment essentiels pour obtenir une lemmatisation de qualité tout en maintenant des temps de traitement raisonnables.

6 Organisation : Répartition des tâches

Nous avons réparti les activités comme suit :

- **Galust** :
  - Découpage des données et nettoyage des corpus (Moses)
  - Gestion et mise en place de la partie "Europarl formes fléchies" et "Europarl lemmatisé" (entraînements et évaluations)
  - Rédaction du rapport (préparation des sections, mise en forme, intégration des tableaux)
- **Mark** :
  - Lemmatization des données (Spacy)
  - Gestion et mise en place de la partie "Europarl + Emea" en formes fléchies et en lemmes (entraînements et évaluations)
  - Mise en ligne du projet sur GitHub, nettoyage et commentaire du code pour la version finale

Références

[1] Klein, G., Kim, Y., Deng, Y., Senellart, J., & Rush, A. M. (2017). *OpenNMT : Open-Source Toolkit for Neural Machine Translation.* ACL 2017. https ://aclanthology.org/P17-4012

[2] *Documentation officielle OpenNMT-py.* https ://opennmt.net/OpenNMT-py/quickstart.html

[3] *Europarl corpus.* http ://opus.nlpl.eu/Europarl.php

[4] *EMEA corpus.* http ://opus.nlpl.eu/EMEA.php

[5] *Moses SMT : Tokenization, nettoyage.* https ://github.com/moses-smt/mosesdecoder

[6] *SacreBLEU.* https ://github.com/mjpost/sacrebleu

[7] Honnibal, M., & Montani, I. (2017). *spaCy 2 : Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing.* https ://spacy.io/